CHAPTER 3 MANAJEMEN MEMORI: VIRTUAL MEMORY

Understanding Operating Systems
Fifth Edition

Pendahuluan

- Evolusi *virtual memory*
  - Paged, demand paging, segmented, segmented/demand paging
- Perbaikan di area:
  - Penyimpananan program secara kontinu
  - Perlunya menyimpan keseluruhan program kedalam memori saat eksekusi
  - Fragmentasi
  - Overhead yang disebabkan oleh relokasi memori

Pendahuluan (continued)

- Aturan pergantian page
  - First-In First-Out
    - Masuk dulu, keluar dulu
  - Least Recently Used
    - Yang bukan terakhir kali digunakan, keluar dulu
    - *Clock replacement* dan *bit-shifting*
  - Mekanisme paging
  - *The working set*
- *Virtual memory*
  - Konsep dan keuntungan
- *Cache memory*
  - Konsep dan keuntungan

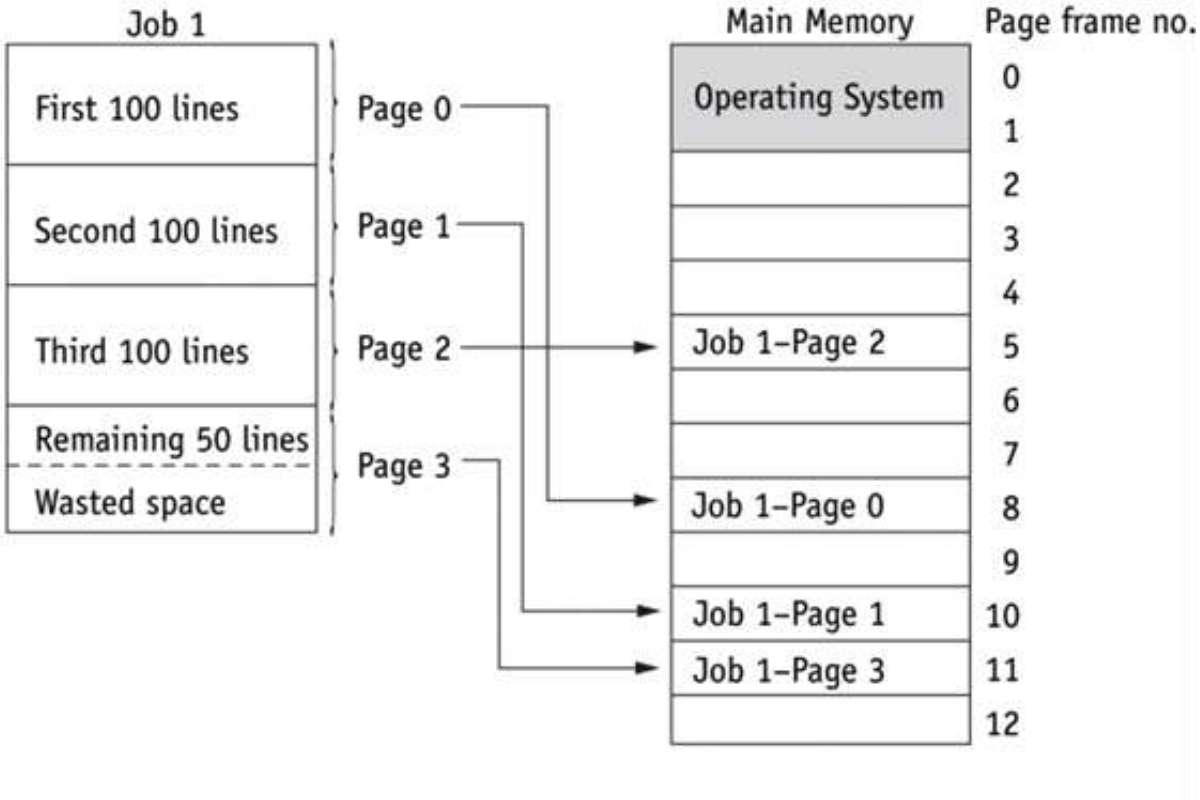
Alokasi Page Memori

- Jobs dibagi kedalam page berukuran sama
- Kondisi terbaik
  - Ukuran page = Ukuran blok memori (page frames) = Ukuran seksi disk (sector, block)
    - Ukuran tergantung pada sistem operasi dan ukuran sektor disk
- Tugas manager memori sebelum eksekusi program
  - Menentukan jumlah page dalam program
  - Menyediakan cukup page frame kosong di memori utama
  - Menyimpan semua page program ke dalam page frame
- Keuntungan menyimpan program secara *non-contiguously*
  - Masalah baru: melacak page-page jobs

Alokasi Page Memori(continued)

Gambar 3.1

Program yang terlalu panjang untuk disimpan dalam satu page, dibagi ke dalam page-page berukuran sama, yang dapat disimpan di page frame yang kosong. Di contoh ini, setiap page frame dapat menyimpan 100 baris. Job 1 panjangnya 350 baris, dibagi menjadi 4 page frame (terdapat fragmentasi dalam di frame terakhir).



Alokasi Page Memori (continued)

- 3 tabel untuk melacak page-page
  - **Job Table (JT)**
    - Ukuran job
    - Lokasi memori dimana PMT-nya disimpan
  - **Page Map Table (PMT)**
    - Jumlah page
    - Alamat memori page frame terkait
  - **Memory Map Table (MMT)**
    - Lokasi tiap page frame
    - Status free/busy

Alokasi Page Memori (continued)

Job Table

Job Size	PMT Location
400	3096
200	3100
500	3150

(a)

Job Table

Job Size	PMT Location
400	3096
500	3150

(b)

Job Table

Job Size	PMT Location
400	3096
700	3100
500	3150

(c)

Tabel 3.1

Di Bagian Job Table ini (a) tadinya memiliki 3 entri untuk tiap job dalam proses. Ketika job kedua selesai (b), entri di table dikosongkan dan digantikan oleh (c) informasi mengenai job selanjutnya yang akan diproses.

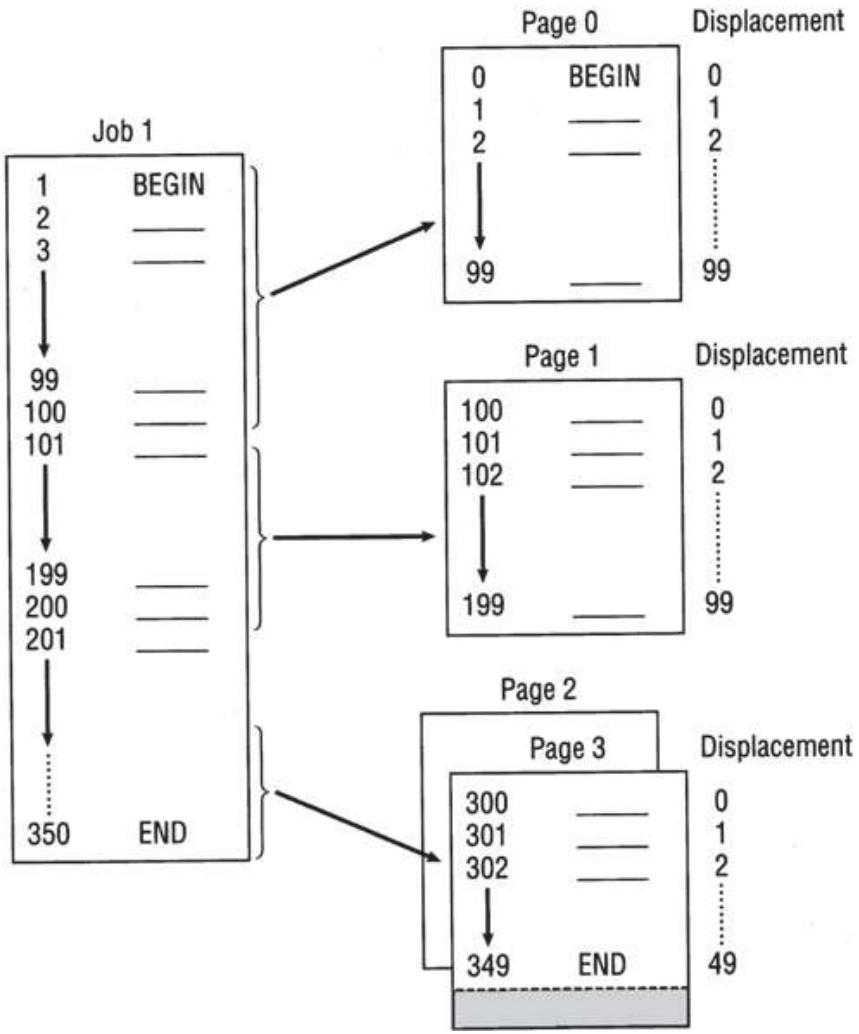
Alokasi Page Memori (continued)

- **Displacement** (offset) sebuah baris
  - Menentukan jarak baris dari awal page
  - Meletakkan baris pada page frame-nya
  - Nilai-nya relatif
- Menentukan jumlah page dan offset baris:
  - Bagi alamat ruang memori job dengan ukuran page
  - **Jumlah page**: Hasil (atas) dari pembagian
  - **Displacement**: sisa pembagian

Paged Memory Allocation (continued)

(figure 3.2)

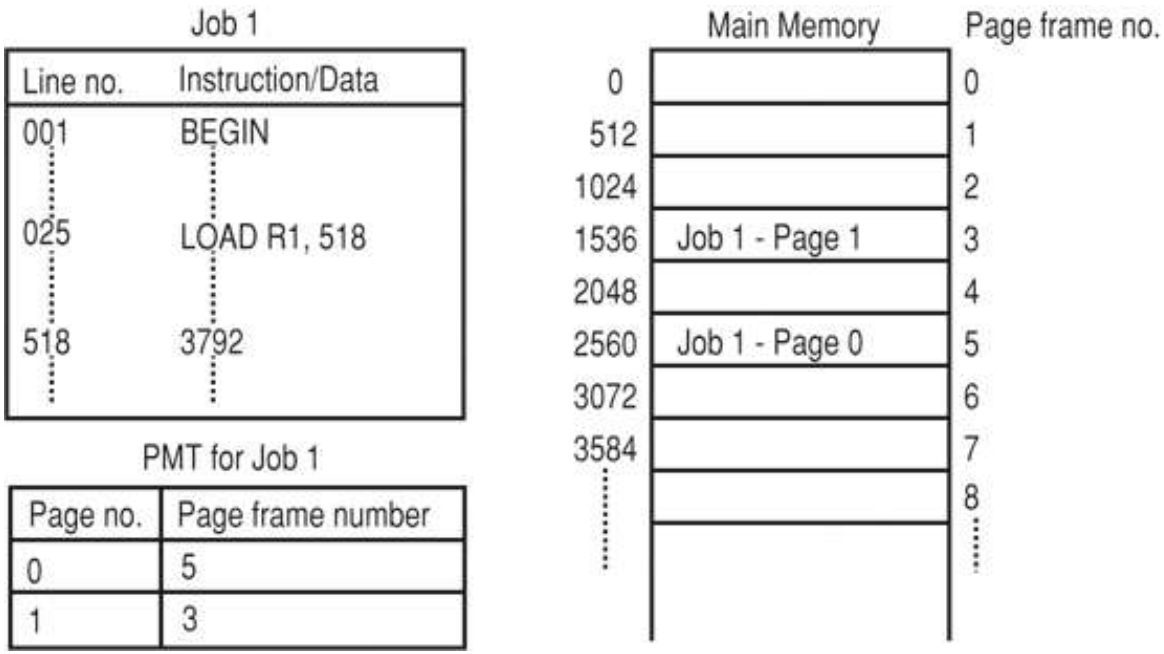
Job 1 is 350 lines long and is divided into four pages of 100 lines each.



Alokasi Page Memori (continued)

- Langkah-langkah untuk menentukan lokasi sebuah baris pada memori
  - Tentukan jumlah page dan offset baris
  - Rujuk ke PMT job nya
    - Tentukan page frame yang berisi page yang dimaksud
  - Dapatkan alamat awal page frame
    - Kalikan jumlah page frame dengan ukuran page frame
  - Tambahkan offset ke alamat awal page frame
- Resolusi alamat
  - Menterjemahkan alamat ruang job ke alamat fisik
  - Alamat relatif ke absolut

Alokasi Page Memori (continued)



(figure 3.3)

Job 1 with its Page Map Table. This snapshot of main memory shows the allocation of page frames to Job 1.

Alokasi Page Memori (continued)

- Keuntungan
  - Memungkinkan alokasi job di memori secara tidak berurutan/*non-contiguous*
    - Penggunaan memori yang efisien
- Kekurangan
  - Meningkatkan overhead untuk resolusi alamat
  - Fragmentasi dalam, di page terakhir
  - Harus menyimpan seluruh job ke memori
- Pemilihan ukuran page sangat krusial
  - Terlalu kecil: PMT menjadi sangat panjang
  - Terlalu besar: fragmentasi dalam terlalu banyak

Demand Paging

- Page dibawa ke memori hanya jika dibutuhkan
  - Tidak keseluruhan program harus disimpan di memori, saat dieksekusi
  - Membutuhkan akses page berkecepatan tinggi
- Menggunakan teknik-teknik pemrograman tertentu
  - Modul-modul ditulis secara sekuensial
    - Page-page tidak selalu diperlukan bersama-sama
  - Misal
    - Modul penanganan eror user

Demand Paging (continued)

- Memungkinkan konsep **virtual memory**
  - Memberikan memori fisik yang tidak terbatas
  - Jobs dijalankan dengan penggunaan memori utama yang lebih sedikit
  - Membutuhkan akses ke media penyimpanan sekunder berkecepatan tinggi
    - Bekerja langsung dengan CPU
  - **Swapping**: bagaimana dan kapan page dikirim/dialihkan ke memori
    - Tergantung aturan awal SO

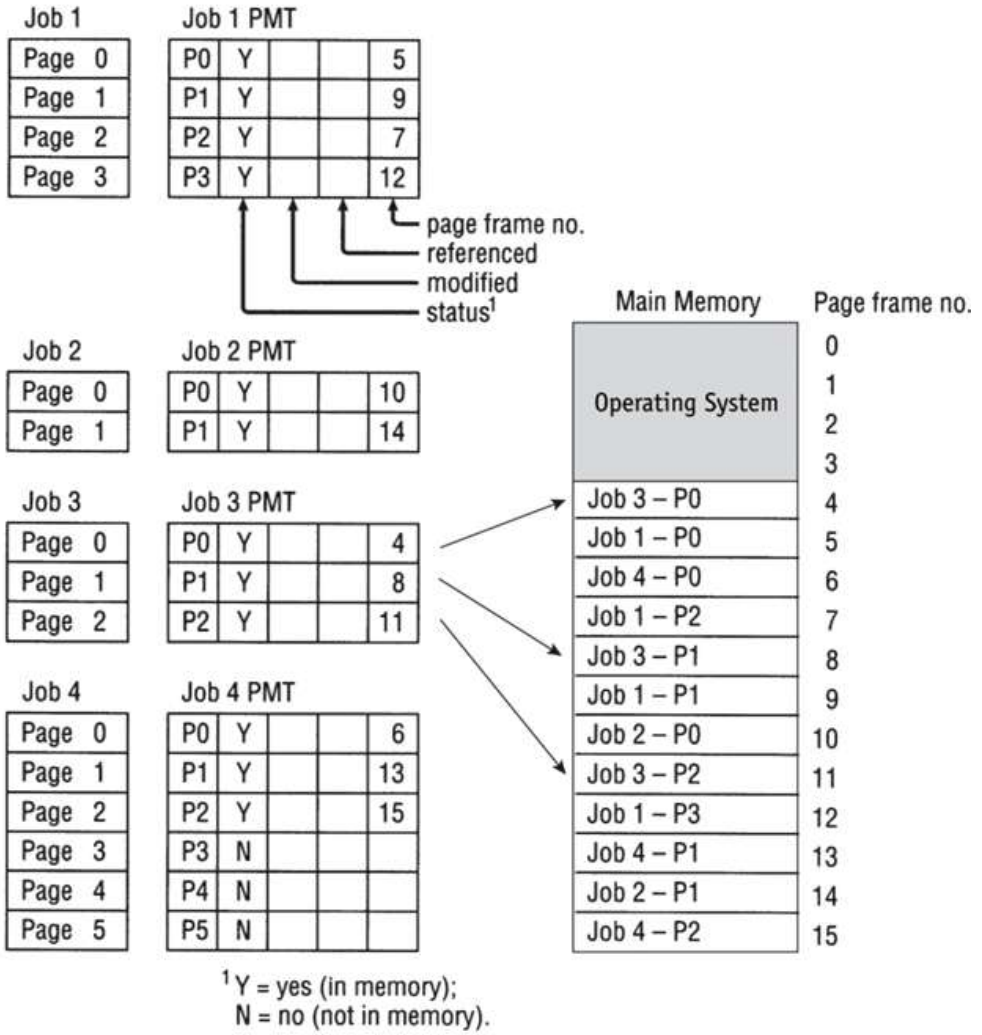
Demand Paging (continued)

- Manajer memori membutuhkan 3 tabel
  - **Job Table**
  - **Page Map Table** memiliki 3 field baru untuk menandai
    - Apakah page yang diminta sudah di dalam memori
    - Apakah isi page sudah diubah
    - Apakah page sudah dirujuk akhir-akhir ini
      - Menentukan page mana yang tetap berada di memori dan mana yang di-*swap*
- **Memory Map Table**

Demand Paging (continued)

Gambar 3.5

Demand paging mengharuskan Page Map Table setiap job melacak setiap page yang disimpan/dihapus dari memori. Setiap PMT melacak status oage, apakah sudah mengalami pengubahan, apakah baru-baru ini dirujuk, dan jumlah page frame untuk setiap page yang saat ini berada di memori. (Catatan: untuk ilustrasi ini PMT disederhanakan. Lihat tabel 3.3 untuk lebih detail.)



Demand Paging (continued)

- **Proses Swapping**
  - Menukar page yang ada di memori utama dengan page yang ada di penyimpanan sekunder / HD.
  - Melibatkan
    - Menyalin isi page yang di memori ke disk (jika telah diubah)
    - Menulis page baru ke page frame yang kosong
  - Membutuhkan interaksi khusus dengan:
    - Komponen-komponen hardware
    - Algoritma perangkat lunak
    - Skema kebijakan/aturan

Demand Paging (continued)

- Pemrosesan instruksi perangkat keras
- Page fault: kegagalan dalam menemukan sebuah page di memori
- Page fault handler
  - Bagian dari SO
  - Menentukan apakah ada page frame kosong di memori
    - Jika iya: page yang diminta disalin ke penyimpanan sekunder
    - Jika tidak: terjadilah swapping

Demand Paging (continued)

- **Thrashing**
  - Terlalu banyak swapping antara memori utama dan penyimpanan sekunder
  - Karena page yang baru saja dihapus dari memori dipanggil kembali tidak lama setelah itu
  - Operasi menjadi tidak efisien
  - Terjadi antar jobs
    - Banyak jobs dalam jumlah besar bersaing untuk page kosong yang jumlahnya relatif sedikit
  - Terjadi dalam sebuah job
    - Di loop antar page

Demand Paging (continued)

- Keuntungan
  - Job tidak lagi dibatasi oleh ukuran fisik memori (konsep virtual memory)
  - Penggunaan memori lebih efisien dibanding skema sebelumnya
  - Respon lebih cepat
- Kekurangan
  - Meningkatkan overhead karena banyaknya tabel dan interupsi page

Konsep dan Aturan Pergantian Page

- Aturan untuk memilih page yang akan dihapus
  - Penting untuk efisiensi sistem
- Aturan pergantian page
  - Aturan **First-In First-Out (FIFO)**
    - Page yang dihapus adalah yang paling lama sudah berada di memori
  - Aturan **Least Recently Used (LRU)**
    - Page yang dihapus adalah yang paling lama tidak diakses
- Mekanisme konsep paging
- The working set concept

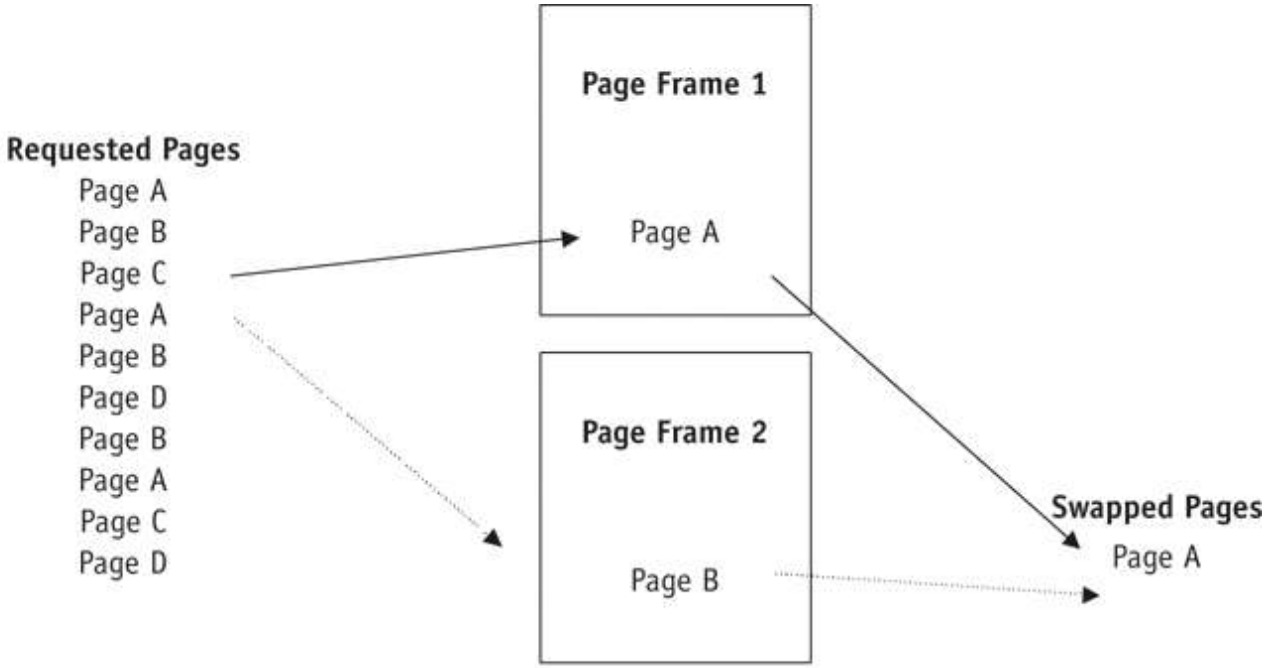
First-In First-Out

- Menghapus page yang paling lama berada di memori
- **Efisiensi**
  - Rasio interupsi page dengan permintaan page
  - Semakin banyak interupsi semakin tidak efisien
  - Contoh FIFO: tidak terlalu baik
    - Efisiensi-nya: 9/11 atau 82%
- **FIFO anomaly**
  - Lebih banyak memori tidak mengakibatkan performa lebih baik

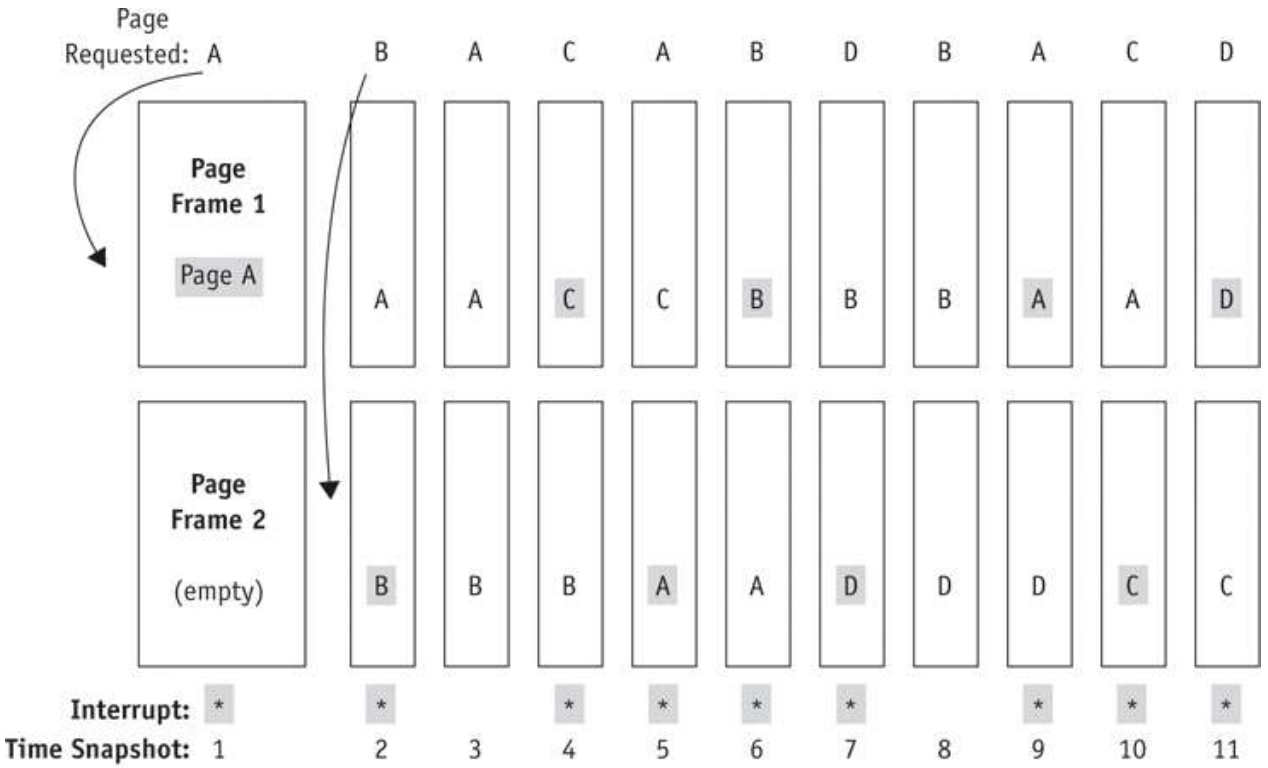
First-In First-Out (continued)

(figure 3.7)

The FIFO policy in action. When the program calls for Page C, Page A must be moved out of the first page frame to make room for it, as shown by the solid lines. When Page A is needed again, it will replace Page B in the second page frame, as shown by the dotted lines. The entire sequence is shown in Figure 3.8.



First-In First-Out (continued)



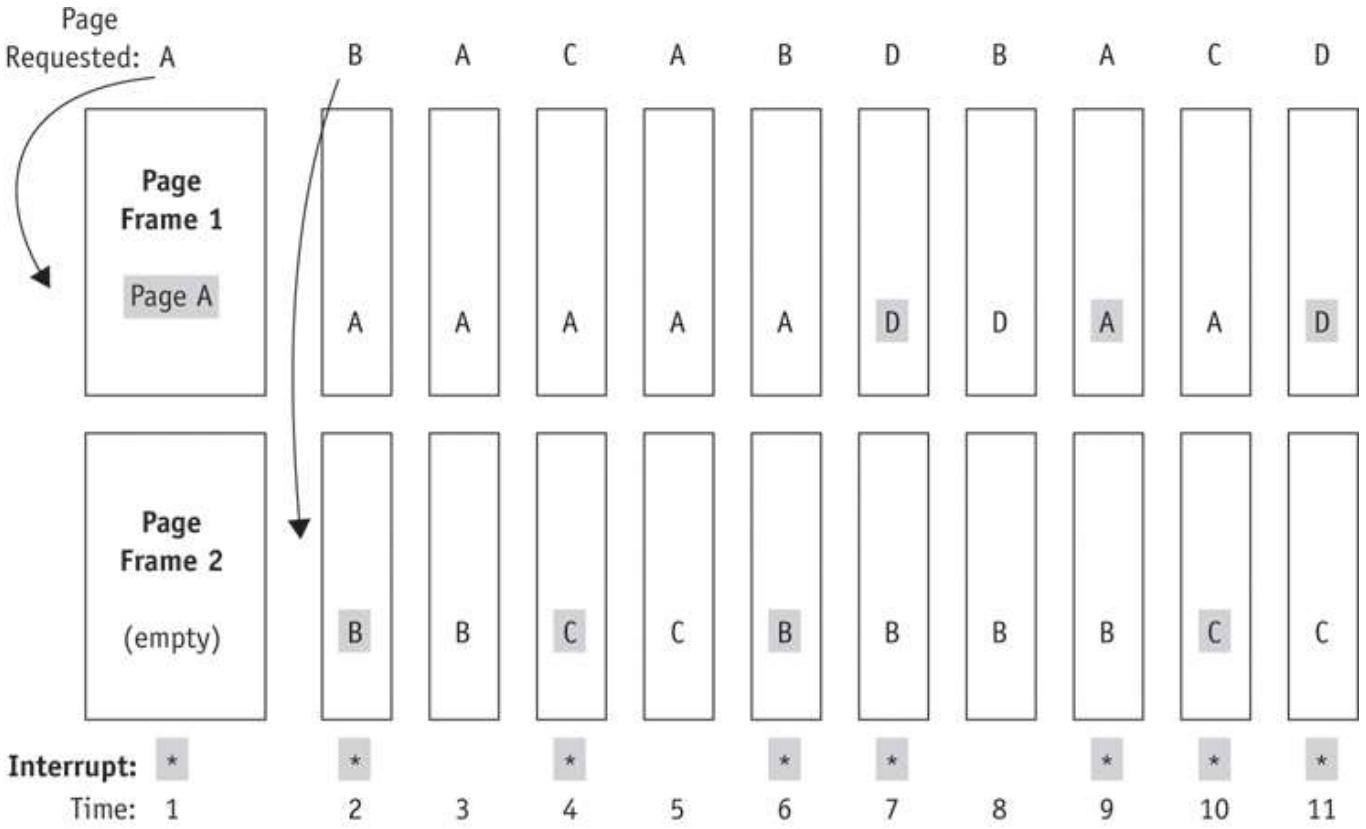
(figure 3.8)

Using a FIFO policy, each page requested is swapped into the two available page frames. When the program is ready to be processed all four pages are in secondary storage. When the program calls a page that isn't already in memory, a page interrupt is issued, as shown by the gray boxes and asterisks. This program resulted in nine page interrupts.

Least Recently Used

- Menghapus page yang paling lama tidak diakses
- Efisiensi
  - Jumlah interupsi menurun atau tetap
  - Agak lebih baik (dibanding FIFO): 8/11 atau 73%
- **LRU** merupakan aturan **stack algorithm**
  - Meningkatkan memori utama akan menyebabkan jumlah interupsi menurun/tetap

Least Recently Used (continued)



(figure 3.9)

Memory management using an LRU page removal policy for the program shown in Figure 3.8. Throughout the program 11 page requests are issued, but they cause only 8 page interrupts.

Least Recently Used (continued)

- 2 variasi
  - Teknik pergantian clock
    - Kecepatan disesuaikan dengan siklus clock komputer
  - Teknik Bit-shifting
    - Menggunakan referensi 8-bit dan teknik bit-shifting
    - Melacak page yang sudah ada di memori

Mekanisme Paging

- Page swapping
  - Manajer memori membutuhkan informasi tertentu
  - Menggunakan informasi dari Page Map Table
    - Status bits: “0” or “1”

Page	Status Bit	Referenced Bit	Modified Bit	Page Frame
0	1	1	1	5
1	1	0	0	9
2	1	0	0	7
3	1	1	0	12

(table 3.3)

Page Map Table for Job 1 shown in Figure 3.5.

Mekanisme Paging (continued)

- Arti bit pada PMT
  - Status bit
    - Menandai apakah page telah berada di memori
  - Referenced bit
    - Menandai apakah page baru-bari ini dirujuk
    - Digunakan oleh LRU untuk menentukan apakah akan diswap / tidak
  - Modified bit
    - Menandai apakah isi telah diubah
    - Digunakan untuk menentukan apakah page harus disalin ke penyimpnanan sekunder sebelum di swap

Mekanisme Paging (continued)

(table 3.4)

The meaning of the bits used in the Page Map Table.

Status Bit		Modified Bit		Referenced Bit	
Value	**Meaning**	**Value**	**Meaning**	**Value**	**Meaning**
0	not in memory	0	not modified	0	not called
1	resides in memory	1	was modified	1	was called

(table 3.5)

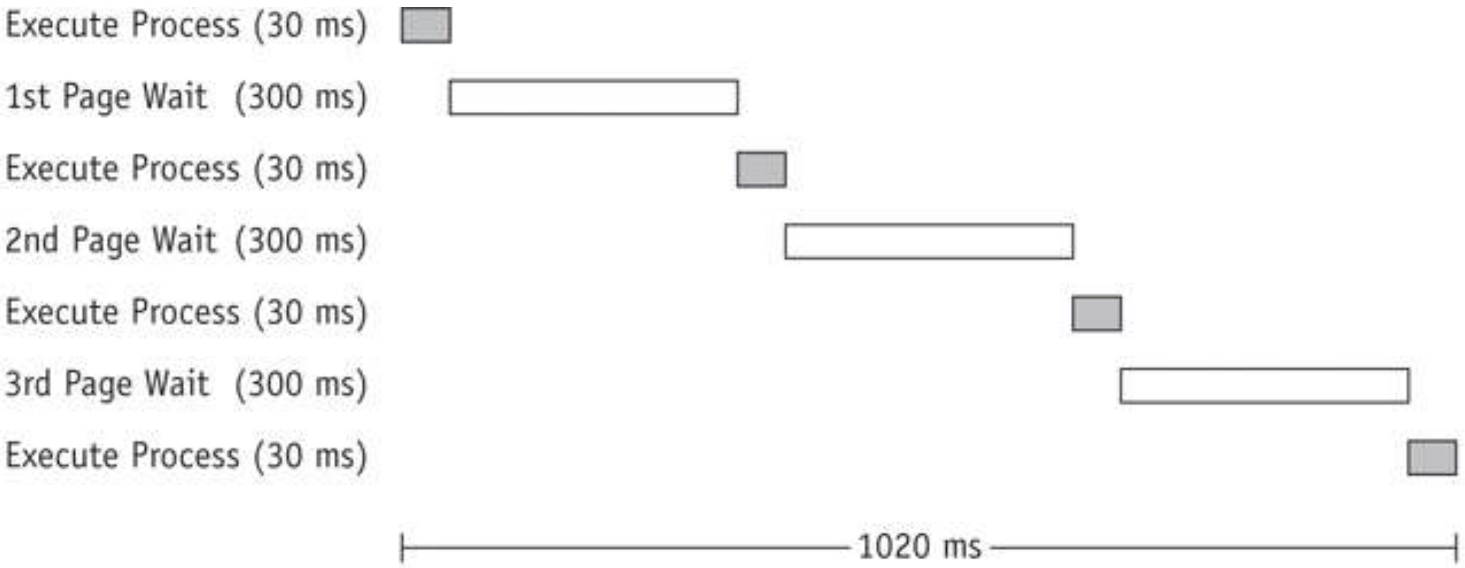
Four possible combinations of modified and referenced bits and the meaning of each.

	Modified	Referenced	Meaning
Case 1	0	0	Not modified AND not referenced
Case 2	0	1	Not modified BUT was referenced
Case 3	1	0	Was modified BUT not referenced [impossible?]
Case 4	1	1	Was modified AND was referenced

The Working Set

- Sekumpulan page yang berada di memori diakses langsung tanpa menimbulkan page fault
  - Meningkatkan kinerja skema demand page
- Membutuhkan konsep “locality of reference”
  - Terjadi di program yang memiliki struktur yang baik
    - Hanya sebagian kecil page yang diperlukan selama eksekusi
- Pertimbangan sistem bagi waktu / time sharing
- Sistem menentukan
  - Jumlah page yang membentuk working set
  - Jumlah maksismum page yang diperbolehkan dalam 1 working set

The Working Set (continued)



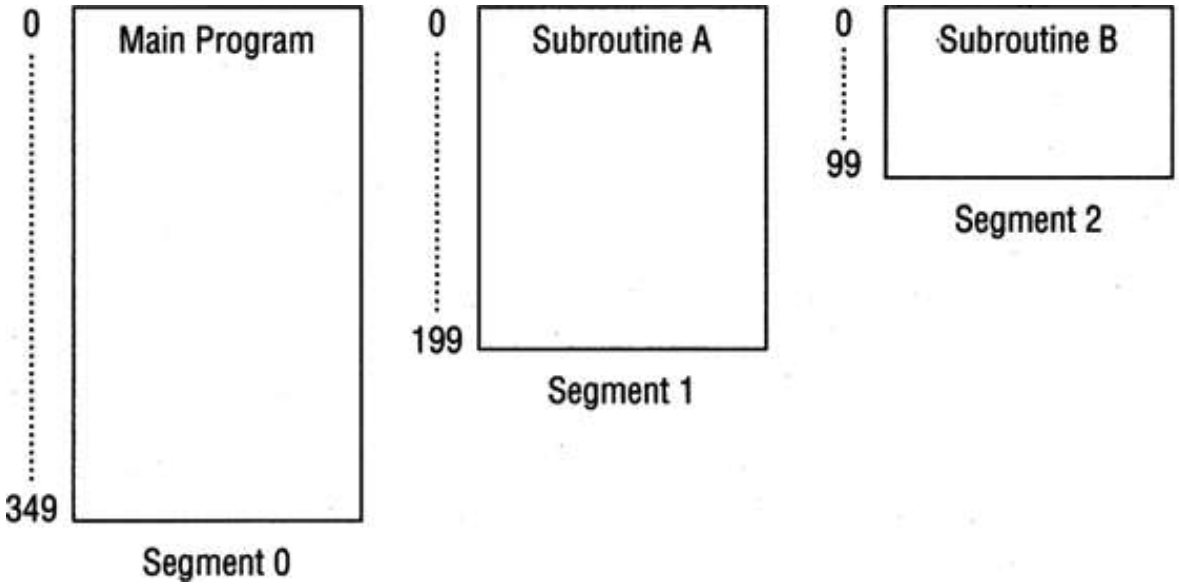
(figure 3.12)

Time line showing the amount of time required to process page faults for a single program. The program in this example takes 120 milliseconds (ms) to execute but an additional 900 ms to load the necessary pages into memory. Therefore, job turnaround is 1,020 ms.

Alokasi Segmen Memori

- Setiap job dibagi menjadi beberapa segmen
  - Segmen-segmen berukuran berbeda
  - Satu dalam setiap modul berisi fungsi yang masih berhubungan
- Mengurangi page faults
- Memori utama tidak lagi dibagi ke dalam page frame
  - Sekarang dialokasikan secara dinamis
- Modul struktur program menentukan segmen
  - Setiap segmen dinomeri ketika di kompilasi/assembled
  - Menghasilkan Segment Map Table (SMT)

Alokasi Segmen Memori (continued)



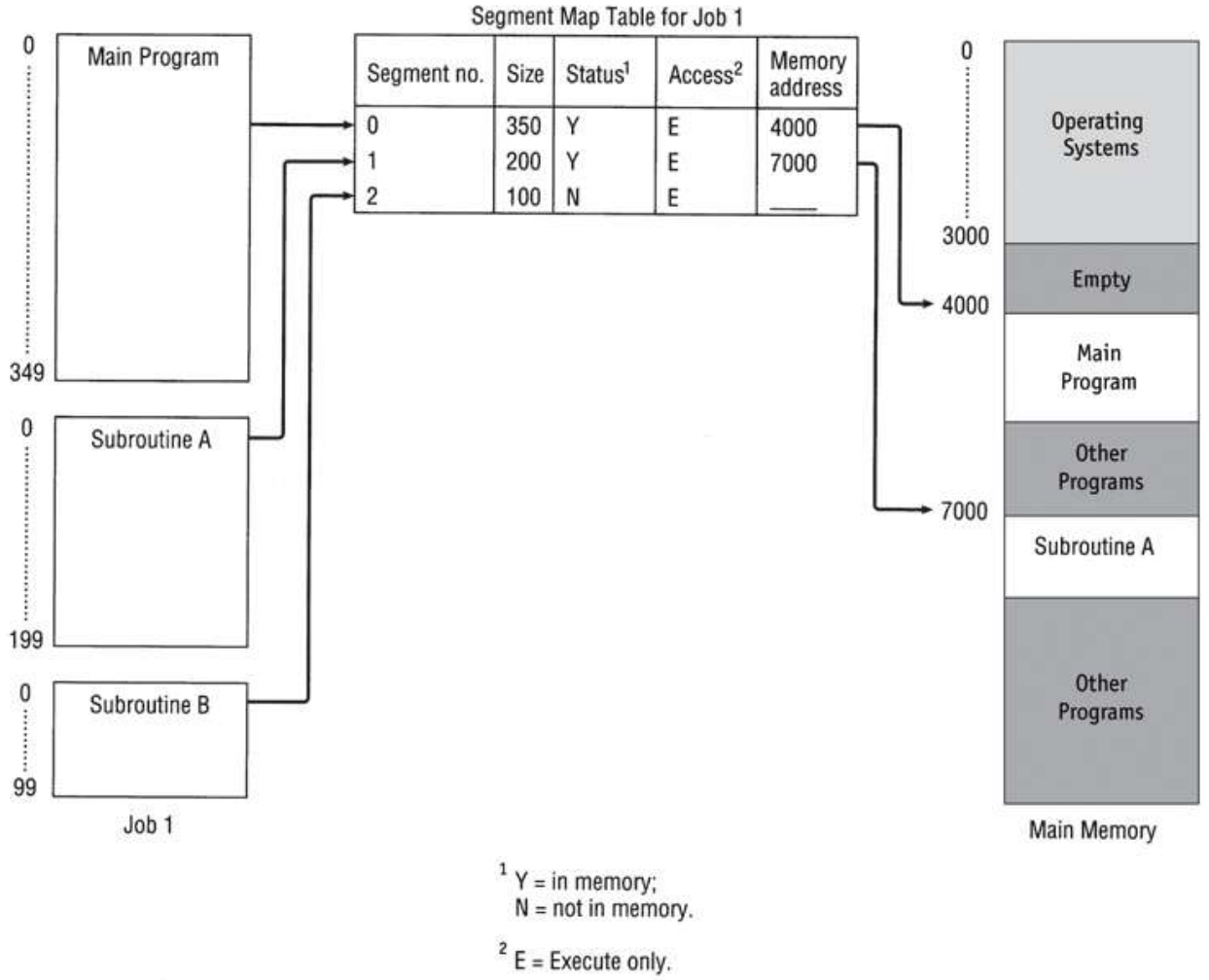
(figure 3.13)

Segmented memory allocation. Job 1 includes a main program, Subroutine A, and Subroutine B. It is one job divided into three segments.

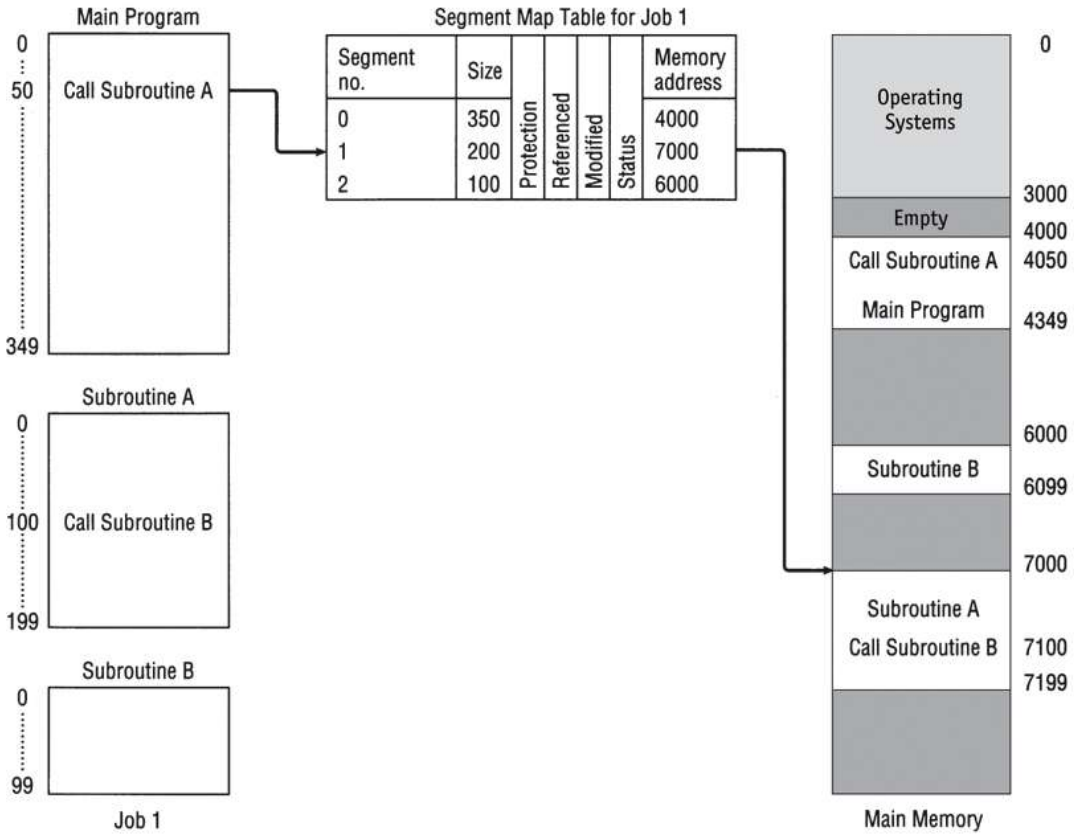
Alokasi Segmen Memori (continued)

(figure 3.14)

The Segment Map Table tracks each segment for Job 1.



Alokasi Segmen Memori (continued)



(figure 3.15)

During execution, the main program calls Subroutine A, which triggers the SMT to look up its location in memory.

Alokasi Segmen Memori (continued)

- Manajer memori melacak segmen menggunakan tabel
  - **Job Table**
    - Daftar semua job dalam proses (satu untuk keseluruhan sistem)
  - **Segment Map Table**
    - Daftar yang berisi detail setiap segmen (satu untuk tiap job)
  - **Memory Map Table**
    - Memonitor alokasi memori utama (satu untuk keseluruhan sistem)
- Instruksi dengan segmen diurutkan secara sekuensial
- Segmen tidak harus disimpan secara berdampingan

Alokasi Segmen Memori (continued)

- Persyaratan skema pengalamatan
  - Jumlah segmen dan displacement
- Keuntungan
  - Tidak ada fragmentasi dalam/internal
  - Memori dialokasikan secara dinamis
- Kekurangan
  - Kesulitan untuk mengatur segmen dengan panjang yang berbeda-beda di penyimpanan sekunder
  - Fragmentasi luar / external

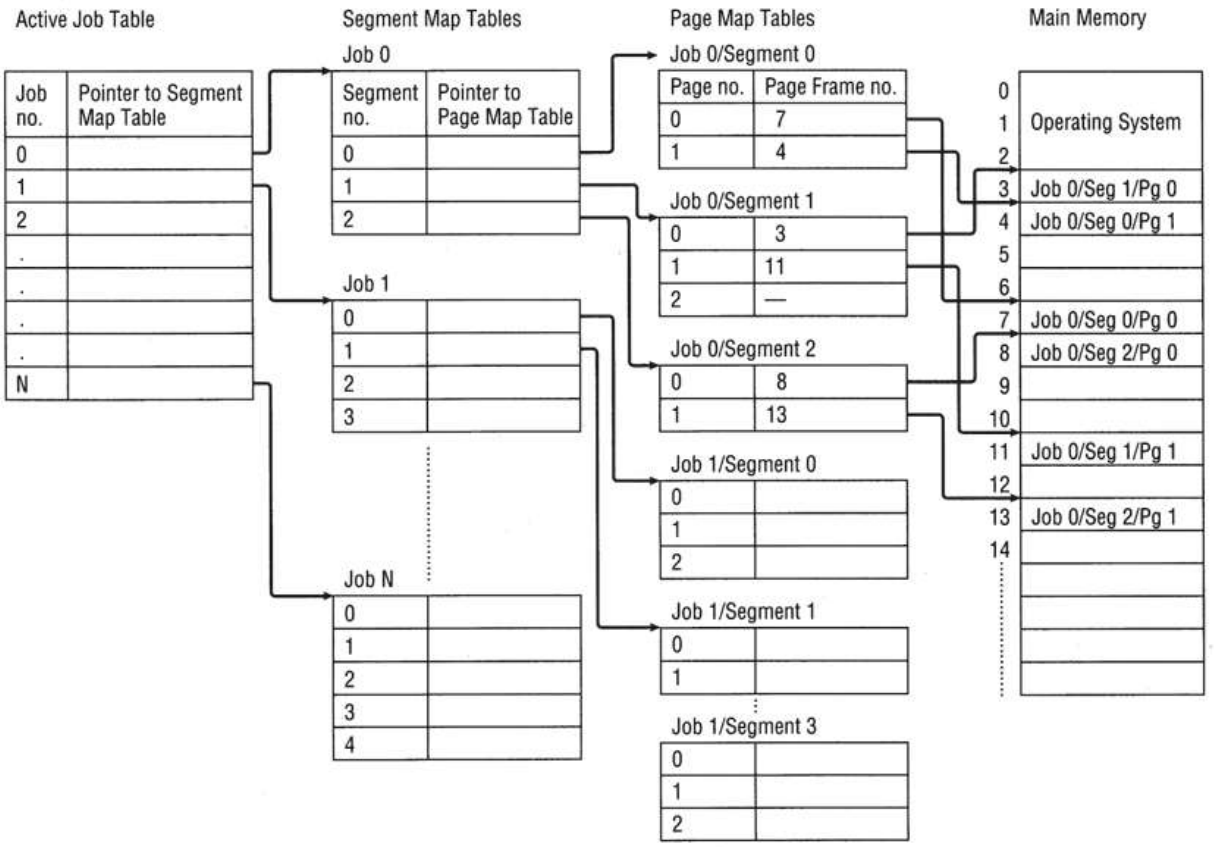
Segmented/Demand Paged Memory Allocation

- Segmen dibagi-bagi lagi menjadi page-page berukuran sama
  - Lebih kecil dari sebagian besar segmen
  - Lebih mudah dimanipulasi dibanding keseluruhan segmen
  - Keuntungan logis dari segmentasi
  - Keuntungan fisik dari paging
- Masalah segmentasi teratasi
  - *Compaction, external fragmentation, secondary storage handling*
- Persyaratan skema pengalamatan
  - Jumlah segmen, jumlah page dalam segmen tersebut, displacement dalam page tersebut

Segmented/Demand Paged Memory Allocation (continued)

- Skema membutuhkan 4 tabel
  - **Job Table**
    - Daftar semua job dalam proses (satu untuk keseluruhan sistem)
  - **Segment Map Table**
    - Daftar berisi detail setiap segmen (satu untuk tiap job)
  - **Page Map Table**
    - Daftar berisi detail setiap page (satu untuk tiap segmen)
  - **Memory Map Table**
    - Memonitor alokasi page frame di main memori (satu untuk keseluruhan sistem)

Segmented/Demand Paged Memory Allocation (continued)



(figure 3.16)

How the Job Table, Segment Map Table, Page Map Table, and main memory interact in a segment/paging scheme.

Segmented/Demand Paged Memory Allocation (continued)

- Keuntungan
  - Ukuran virtual memori besar
  - Segment di load berdasarkan permintaan
  - Mendapatkan keuntungan logika (pemrograman) dari segmentasi
  - Mendapatkan keuntungan fisik dari paging
- Kekurangan
  - Overhead untuk menangani tabel
  - Memori yang dibutuhkan untuk tabel segmen dan page

Virtual Memory

- Memungkinkan eksekusi program walaupun tidak seluruhnya disimpan ke dalam memori
- Membutuhkan kerjasama antara manajer memori dan perangkat keras prosesor
- Keuntungan
  - Ukuran job tidak dibatasi oleh ukuran memori utama
  - Memori utama digunakan secara lebih efisien
  - Memungkinkan jumlah multiprogramming tidak terbatas
  - Menghilangkan fragmentasi eksternal dan meminimalkan fragmentasi internal

Virtual Memory (continued)

- Keuntungan (continued)
  - Memungkinkan sharing kode dan data
  - Memfasilitasi linking dinamis untuk segmen-segmen program
- Kekurangan
  - Meningkatkan biaya untuk perangkat lunak prosesor
  - Meningkatkan overhead untuk menangani interupsi page
  - Meningkatkan kompleksitas perangkat lunak untuk menghindari *thrashing*

Virtual Memory (continued)

(table 3.6)

Comparison of virtual memory with paging and segmentation.

Virtual Memory with Paging	Virtual Memory with Segmentation
Allows internal fragmentation within page frames	Doesn't allow internal fragmentation
Doesn't allow external fragmentation	Allows external fragmentation
Programs are divided into equal-sized pages	Programs are divided into unequal-sized segments that contain logical groupings of code
Absolute address calculated using page number and displacement	Absolute address calculated using segment number and displacement
Requires PMT	Requires SMT

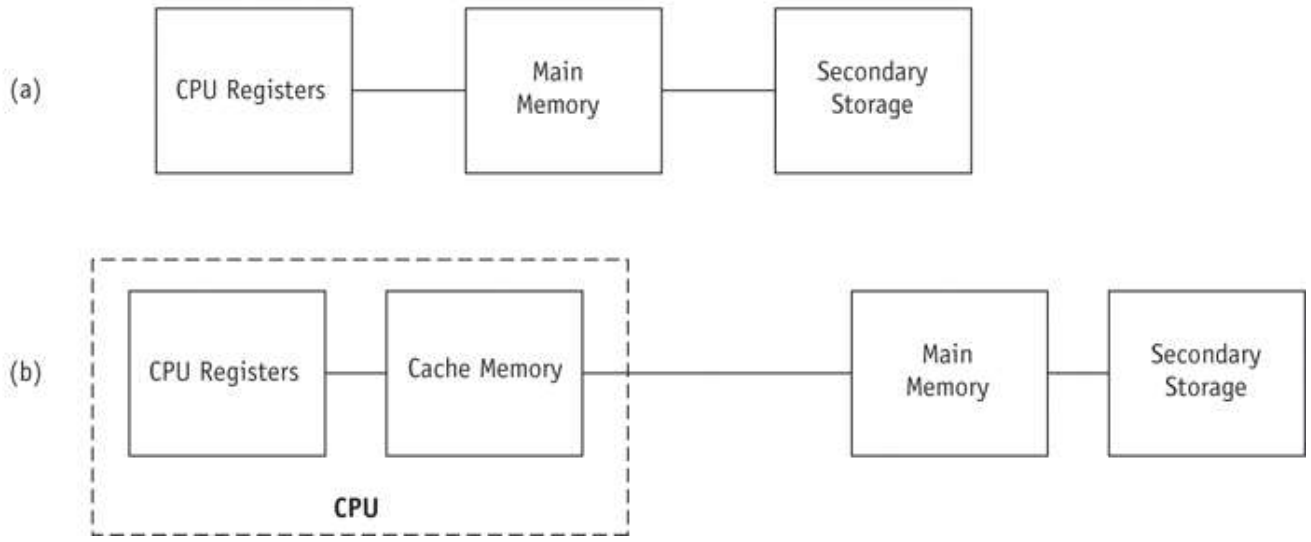
Cache Memory

- Unit memori intermediate yang berukuran kecil dan berkecepatan tinggi
- Kinerja sistem komputer meningkat
  - Waktu untuk akes memori berkurang jauh
  - Akses prosesor lebih cepat dibanding ke memori utama
  - Menyimpan data dan instruksi yang sering dipakai
- 2 level cache
  - L2: terhubung ke CPU; berisi salinan bus data
  - L1: dibangun sepasang ke dalam CPU; menyimpan instruksi dan data
- Data/instruksi pindah dari memori utama ke cache
  - Menggunakan metode yang sama dengan algoritma paging

Cache Memory (continued)

(figure 3.17)

Comparison of (a) traditional path used by early computers between main memory and the CPU and (b) path used by modern computers to connect the main memory and the CPU via cache memory.



Rangkuman

- Alokasi page memori
  - Penggunaan memori yang efisien
  - Mengalokasikan job-job ke lokasi memori yang tidak berdampingan
  - Masalah
    - Meningkatkan overhead
    - Fragmentation internal
- Skema demand paging
  - Menghilangkan batasan ukuran fisik memori
  - LRU meningkatkan efisiensi (dibanding FIFO)
- Skema alokasi segmen memori
  - Mengatasi masalah fragmentasi internal

Rangkuman (continued)

- Segmented/demand paged memory
  - Masalah yang diatasi
    - *Compaction, external fragmentation, secondary storage handling*
- Virtual memory
  - Program dieksekusi tanpa harus seluruhnya di store ke dalam memori
  - Ukuran jobs tidak dibatasi ukuran fisik memori
- Cache memory
  - CPU dapat mengeksekusi instruksi lebih cepat

Perbandingan Skema Alokasi Memori (Bab 2 & 3)

Skema	Masalah yang diatasi	Masalah yang timbul	Perubahan pada Software
Single user contiguous		Ukuran jobs terbatas pada ukuran fisik memori; CPU sering idle	Tidak ada
Fixed partition	Waktu idle CPU	Fragmentasi internal; ukuran job dibatasi oleh ukuran partisi	Penambahan penjadwalan prosesor; penambahan penanganan proteksi
Dynamic partition	Fragmentasi internal	Fragmentasi eksternal	Tidak ada
Relocatable dynamic partition	Fragmentasi internal	Overhead untuk compaction; ukuran job dibatasi ukuran fisik memori	Algoritma untuk compaction
Paged	Kebutuhan untuk compaction	Memori yang dibutuhkan untuk tabel; ukuran jobs dibatasi oleh ukuran fisik memori; fragmentasi internal	Algoritma untuk menangani PMT
Demand paged	Ukuran jobs tidak lagi dibatasi oleh ukuran fisik memori; penggunaan memori lebih efisien; memungkinkan multiprogramming dan time-sharing skala besar	Jumlah tabel lebih banyak; kemungkinan thrashing; overheard yang diperlukan untuk interupsi page; hardware yang diperlukan	Algoritma pergantian page; algoritma pencarian page dan penyimpanan sekunder
segmented	Fragmentasi internal	Kesulitan menangani segmen berukuran bervariasi pada penyimpanan sekunder; fragmentasi eksternal	Dynamic linking package; skema pengalamatan 2 dimensi
Segmented/deman paged	Virtual memori besar; segmen di load berdasarkan permintaan	Overhead untuk tabel; memori yang diperlukan untuk tabel page dan segmen	Skema pengalamatan 3 dimensi